Max-Sum Inference Algorithm

Sargur Srihari
srihari@cedar.buffalo.edu

The max-sum algorithm

- Sum-product algorithm
  - Takes joint distribution expressed as a factor graph
  - Efficiently finds marginals over component variables
- Max-sum addresses two other tasks
  1. Setting of the variables that has the highest probability
  2. Find value of that probability
- Algorithms are closely related
  - Max-sum is an application of dynamic programming to graphical models

Finding latent variable values having high probability

- Consider simple approach
  - Use sum-product to obtain marginals $p(x_i)$ for every variable x_i
  - For each variable find value x_i^* that maximizes marginal
- This would give set of values that are individually most probable
- However we wish to find vector $x^{\max}$ that maximizes joint distribution, i.e.

$$x^{\max} = \arg_x \max p(x)$$

- With join probability $p(x^{\max}) = \max_x p(x)$

Example

- Maximum of joint distribution
  - Occurs at $x=1, y=0$
  - With $p(x=1,y=0)=0.4$
- Marginal $p(x)$
  - $p(x=0) = p(x=0,y=0)+p(x=0,y=0)=0.6$
  - $p(x=1) = p(x=1,y=0)+p(x=1,y=1)=0.4$
- Marginal $p(y)$
  - $P(y=0)=0.7$
  - $P(y=1)=0.3$
- Marginals are maximized by $x=0$ and $y=0$ which corresponds to 0.3 of joint distribution
- In fact, set of individually most probable values can have probability zero in joint

$p(x,y)$	$x=0$	$x=1$
$y=0$	0.3	0.4
$y=1$	0.3	0.0

Max-sum principle

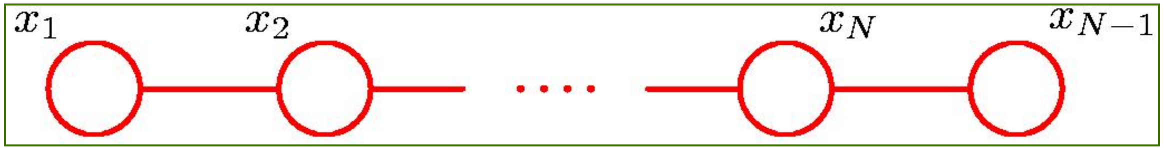
- Seek efficient algorithm for
  - Finding value of x that maximizes $p(x)$
  - Find value of joint distribution at that x
- Second task is written

$$\max_x p(x) = \max_{x_1} \ldots \max_{x_M} p(x)$$

where M is total number of variables

- Make use of distributive law for $\max$ operator
  - $\max(ab,ac) = a \max(bc)$
  - Which holds for $a \geq 0$
  - Allows exchange of products with maximizations

Chain example



- Markov chain joint distribution has form

$$p(x) = \frac{1}{Z}\psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\ldots\psi_{N-1,N}(x_{N-1},x_N)$$

- Evaluation of probability maximum has form

$$\max_x p(x) = \frac{1}{Z}\max_{x_1}\ldots\max_{x_N}\boxed{\psi_{1,2}(x_1,x_2)}\psi_{2,3}(x_2,x_3)\ldots\psi_{N-1,N}(x_{N-1},x_N)$$

- Exchanging max and product operators

$$\max_x p(x) = \frac{1}{Z}\max_{x_1}\left[\psi_{1,2}(x_1,x_2)\left[\ldots\max_{x_N}\psi_{N-1,N}(x_{N-1},x_N)\right]\right]$$

  – Results in
    - More efficient computation
    - Interpreted as messages passed from node x_N to node x_1

Generalization to tree factor graph

- Substitution factored graph expansion

$$p(x) = \prod_s f_s(x_s)$$

- Into $\max_x p(x) = \max_{x_1} \ldots \max_{x_M} p(x)$
- And exchanging maximizations with products
- Final maximization is performed over product of all messages arriving at the root node
- Could be called the *max-product* algorithm

Use of log probabilities

- Products of probabilities can lead to numerical underflow problems
- Convenient to work with logarithm of joint distribution
- Has the effect of replacing products in max-product algorithm with sums
- Thus we obtain the *max-sum* algorithm

Message Passing formulation

- In sum-product we had

From factor node to variable node

$$\mu_{f \to x}(x) = \sum_{x_1} .. \sum_{x_M} f(x, x_1, \ldots, x_M) \prod_{m \in ne(f) \backslash x} \mu_{x_m \to f}(x_m)$$

From variable Node to factor node

$$\mu_{x \to f}(x) = \prod_{l \in ne(x) \backslash f} \mu_{f_l \to x}(x)$$

Initial messages sent by leaf nodes

$$\mu_{x \to f}(x) = 1$$
$$\mu_{f \to x}(x) = f(x)$$

- By replacing sum with max and products with sums of logarithms

$$\mu_{f \to x}(x) = \max_{x_1, .. x_M} \left[\ln f(x, x_1, .. x_M) + \sum_{m \in ne(f) \backslash x} \mu_{x_m \to f}(x_m) \right]$$

$$\mu_{x \to f}(x) = \sum_{l \in ne(x) \backslash f} \mu_{f_l \to x}(x)$$

Initial messages sent by leaf nodes

$$\mu_{x \to f}(x) = 0$$
$$\mu_{f \to x}(x) = \ln f(x)$$

Maximum compution

- At root node in sum-product algorithm

$$p(x) = \prod_{s \in ne(x)} \mu_{f_s \to x}(x)$$

- By analogy in max-sum algorithm

$$p^{\max} = \sum_{s \in ne(x)} \mu_{f_s \to x}(x)$$

Finding variable configuration with maximum value

- In evaluating p^{max} we will also get x^{max} for the most probable value for the root node as

$$x^{\max} = \arg\max_{x} \sum_{s \in ne(x)} \mu_{f_s \to x}(x)$$

- It is tempting to apply the above to from the root back to leaves
  - However there may be multiple configurations of x all of which give rise to maximum value of $p(x)$
    - Recursively repeated at every node
  - So over all configuration need not be the one that maximizes

Modified message passing

- Different type of message passing from the root node to the leaves
- Keeping track of which values of the variables give rise to the maximum state of each variable
- Storing quantities given by

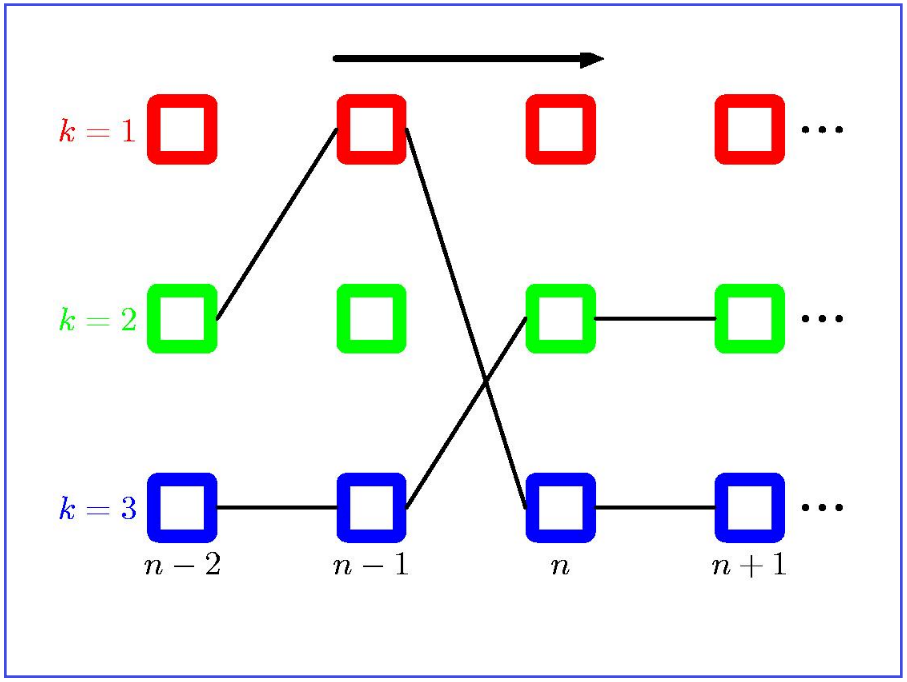
$$\varphi(x_n) = \arg\max_{x_{n-1}}[\ln f_{n\text{-}1,n}(x_{n-1},x_n) + \mu_{x_{n-1}\to f_{n-1,n}}(x_n)]$$

- Understood better by looking at lattice or trellis diagram

Lattice or Trellis Diagram

- $k=2$ and $k=3$ each represent possible values of x_N^{max}
- Two paths give global maximum
  - Can be found by tracing back along opposite direction of arrow

Not a graphical model
Columns represent variables
Row represent states of variable



Backtracking in Trellis

- For each state of given variable there is a unique state of the previous variable that maximizes probability
  - ties are broken systematically or randomly
- Equivalent to propagating a message back down the chain using

$$x_{n-1}^{\max} = \phi(x_n^{\max})$$

- Know as backtracking

Extension to general tree graphs

- Method is generalizable to tree-structured factor graphs
- If a message is sent from a factor node f to a variable node x
  - Maximization is performed over all other variable nodes $x_1,..,x_N$ that are neighbors of the factor node
- Keeping track of which values of the variables gave the maximum

Viterbi Algorithm

- Max-sum algorithm gives exact maximizing configuration for variables provided factor graph is a tree
- Important application is in finding most probable sequence of hidden states in a HMM
  - known as the *Viterbi algorithm*

Max sum versus ICM

- ICM is simpler
- Max sum finds global maximum for tree graphs
- ICM is not guaranteed to find global maximum

Exact inference in general graphs

- Sum-product and max-sum algorithms
  - are efficient and exact solutions
    - to inference problems in tree-structured graphs
- In some cases we need to deal with graphs with loops
- Message passing framework can be generalized to arbitrary graph topologies
- Know as *junction tree* algorithm

Junction Tree Algorithm

- Triangulation:
  - Find chord-less Cycles such as ACBDA and add links such as AB or CD
- Join tree
  - Nodes correspond to maximal cliques of triangulated graph
  - Links connect pairs of cliques that have variables in common
  - Done so as to give a maximal spanning tree defined as
    - Weight of the tree is maximum
    - Weight is sum of weights for links
- Junction tree
  - Tree is condensed so that any clique that is a subset of another clique is absorbed
- Tow-stage message passing algorithm
  - equivalent to sum-product, can be applied to junction tree
  - to find marginals and conditionals